Making a Fortran program for solving a second order linear differential equation

Written by Evgenii Neumerzhitckii

Aug 15, 2019

Contents

Abstract

We write a Fortran program that solves $\ddot{x} - x = 0, x(0) = 1, \dot{x}(0) = 0$ problem numerically using a finite-difference method. We plot the numerical solution and compare it with the exact solution. We look at absolute errors of numerical solutions and investigate their dependence on the size of the time step. We find that global errors of our numerical method are proportional to the square of the time step for time steps between 0.0005 and 1. Using time steps smaller than 0.0005 results in increase of the errors.

The ODE

We want to use a finite-difference method in order to numerically solve the second order linear homogeneous ordinary differential equation

$$\ddot{x} + x = 0, \tag{1}$$

given the initial conditions

$$x(0) = 1 \tag{2}$$

$$\dot{x}(0) = 0. \tag{3}$$

Finding exact solution

We will first find the general solution to Equation 1. The corresponding characteristic equation is

$$\lambda^2 + 1 = 0$$

with roots

$$\lambda = \pm i.$$

Therefore, the general solution to Equation 1 is

$$x = A\cos t + B\sin t, \quad A, B \in \mathbb{R}. \tag{4}$$

Next, we use the initial conditions to find the values of constants A and B. Using Equation 2 gives

$$\begin{aligned} x(0) &= 1 \\ &= A\cos 0 + B\sin 0 \\ &= A. \end{aligned}$$

Therefore,

$$A = 1.$$

Next, we calculate the derivative of x using Equation 4 and $A = 1$:

$$\dot{x} = -\sin t + B\cos t.$$

Applying condition from Equation 3 gives

$$\begin{aligned}\dot{x}(0) &= 0 \\ &= -\sin 0 + B\cos 0 \\ &= B.\end{aligned}$$

We have found that

$$B = 0.$$

Finally, we substitute the values of A and B back into Equation 4 and find the unique solution we desired:

$$\boxed{x = \cos t.}$$

Deriving a recurrence relation

Next, we want to find a recurrence relation for Equation 1 that will be used to solve the equation numerically. We begin by writing an approximation of the second derivative:

$$\ddot{x} \approx \frac{x(t + \Delta t) - 2x(t) + x(t - \Delta t)}{\Delta t^2}.$$

Solving for $x(t + \Delta t)$ gives

$$x(t + \Delta t) \approx -x(t - \Delta t) + 2x(t) + \Delta t^2 \ddot{x} \tag{5}$$

Next, we use approximations

$$\begin{aligned}x_{i-1} &\approx x(t - \Delta t) \\ x_i &\approx x(t) \\ x_{i+1} &\approx x(t + \Delta t), \quad \text{for } i = 0, 1, \ldots, n_t,\end{aligned}$$

where n_t is the number of time steps. With the approximations Equation 5 becomes

$$\begin{aligned}x_{i+1} &= -x_{i-1} + 2x_i + \Delta t^2 \ddot{x}_i \\ &= -x_{i-1} + 2x_i - \Delta t^2 x_i && \text{(Use Equation 1)} \\ &= -x_{i-1} + x_i(2 - \Delta t^2).\end{aligned}$$

Finally, we replace i with $i + 1$ in order to make the indexes start from 0 and not from -1 and get the recurrence relation we wanted:

$$\boxed{x_{i+2} = -x_i + x_{i+1}(2 - \Delta t^2),\ i = 0, 1, \ldots, n_t} \tag{6}$$

Calculating x_1

We have found out recurrence relation but we can't use it yet. We know from Equation 2 that $x_0 = 1$, but we don't know x_1. In order to estimate x_1, we use a power series expansion around $t = 0$:

$$x(0 + \Delta t) \approx x(0) + \dot{x}(0)\Delta t + \frac{1}{2}\ddot{x}(0)\Delta t^2.$$

Next, we substitute $\ddot{x}$ from Equation 1, as well as x and $\dot{x}$ from Equations 2 and 3:

$$x(\Delta t) \approx 1 - \frac{1}{2}\Delta t^2$$

Finally, we substitute the approximation

$$x_1 \approx x(\Delta t)$$

and find the expression for x_1 that we wanted

$$\boxed{x_1 = 1 - \frac{1}{2}\Delta t^2.}$$

Calculating full numerical solution

Next, we write a Fortran program that solves Equation 1 numerically. The program is given the size of the time step Δt, as well as the final value of t. The starting value of t is set to be zero by the initial value conditions. The first two values of x are

$$x_0 = 1$$

$$x_1 = 1 - \frac{1}{2}\Delta t^2.$$

The remaining values are calculated iteratively using the recurrence relation from Equation 6.

Instructions for compiling and running the program are located in the README.md file that comes with the source code.

Plotting approximate and exact solutions

We plot the approximate and exact solutions of Equation 1 on Figure 1 for $\Delta t = 1$. We can see that this time step does not result in good agreement between two solutions. Next, we decrease the time step to $\Delta t = 0.1$, the results are shown on Figure 2. Now the approximate solution looks indistinguishable from the exact one on this scale.

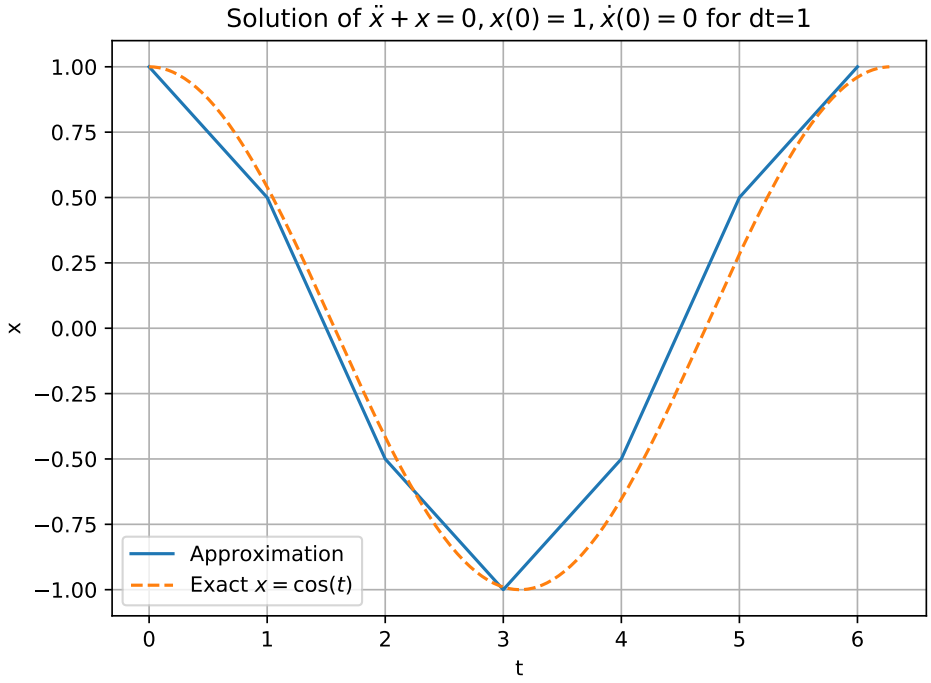

Figure 1: Approximate and exact solutions of the problem $\ddot{x} - x = 0, x(0) = 1, \dot{x}(0) = 0$ for $\Delta t = 1$.

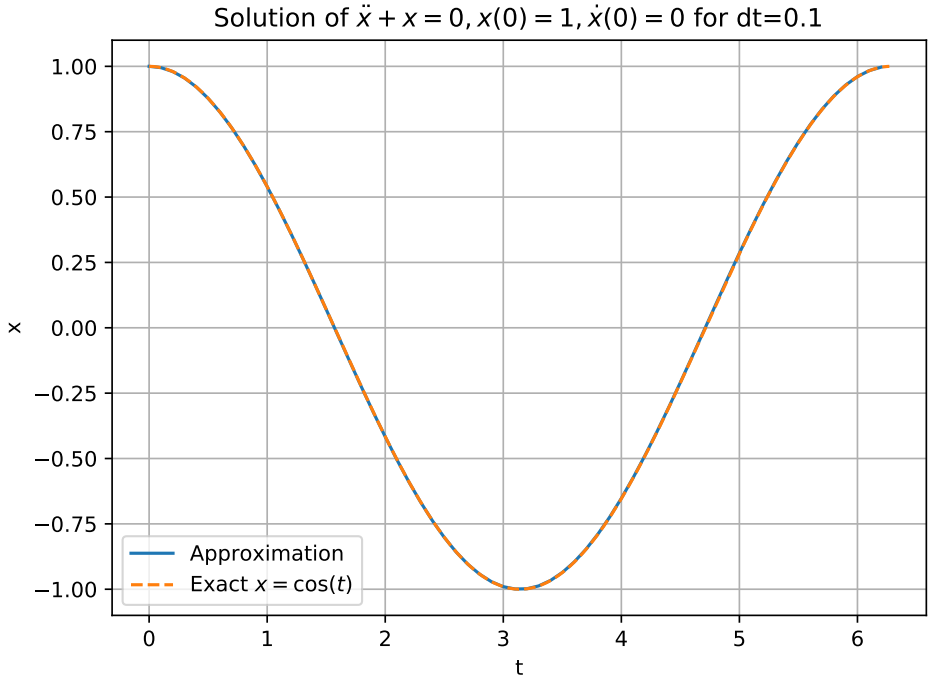

Figure 2: Approximate and exact solutions of the problem $\ddot{x} - x = 0, x(0) = 1, \dot{x}(0) = 0$ for $\Delta t = 0.1$.

Plotting error dependence on t

Next, we want to see how absolute errors – an absolute value of the difference between the approximate and exact solution at each t – depend on t. The absolute errors for $\Delta t = 1$ and $\Delta t = 0.1$ are shown on Figures 3 and 4. We can see that globally the absolute error increases with t (we ignore local periodic oscillations) for both time steps. We can also see that absolute errors from $\Delta t = 1$ solution are about 100 times larger than those from $\Delta t = 0.1$ solution.

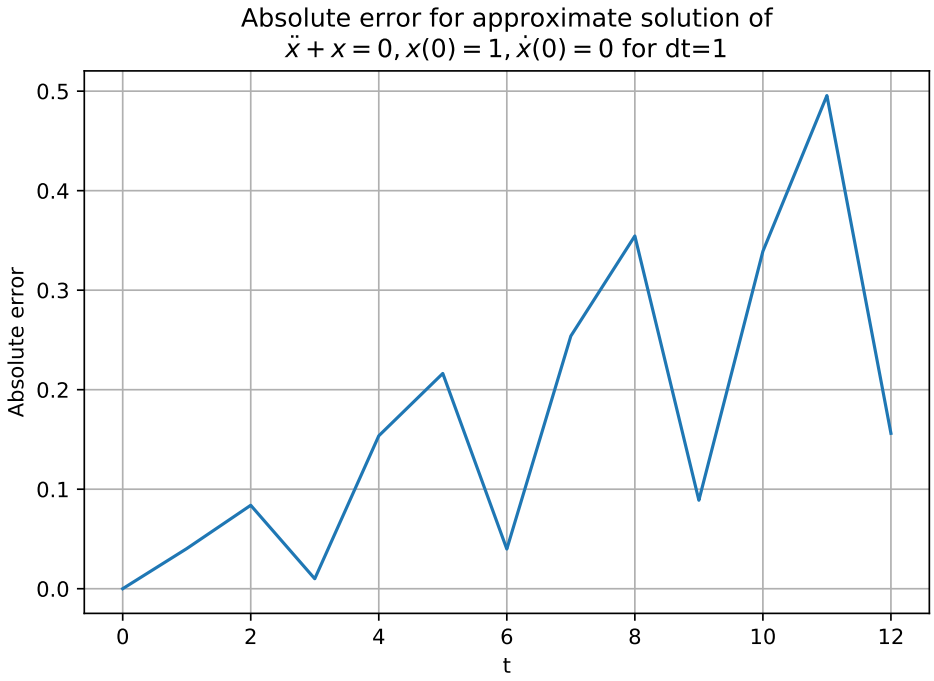

Figure 3: Absolute error of numerical solution of the problem $\ddot{x} - x = 0, x(0) = 1, \dot{x}(0) = 0$ for $\Delta t = 1$.

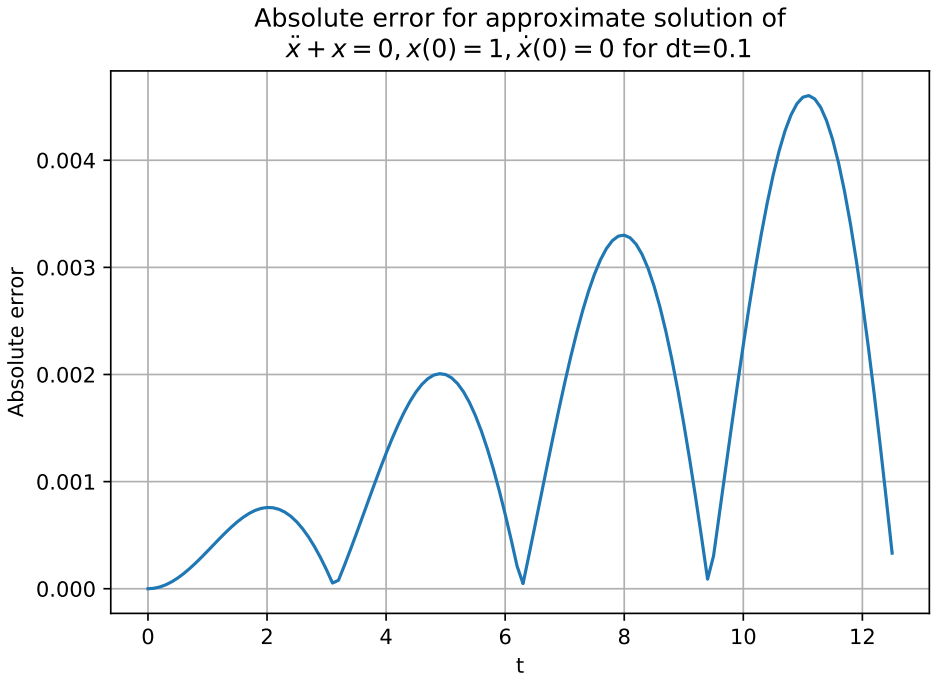

Figure 4: Absolute error of numerical solution of the problem $\ddot{x} - x = 0, x(0) = 1, \dot{x}(0) = 0$ for $\Delta t = 0.1$.

Dependence of errors on Δt

Finally, we want to see how the choice of the time step affects the absolute errors. We run our Fortran program repeatedly using the following values of time steps:

$$\Delta t = 1, 0.5, 0.2, 0.1, 0.05, 0.02, 0.01, 0.005, 0.002, 0.001, 0.0005.$$

Next we calculate the absolute errors of the numerical solutions for different time steps at $t = 11$. The value $t = 11$ was chosen because it corresponds to a large error on Figure 4. The log-log plot of the errors is shown on Figure 5.

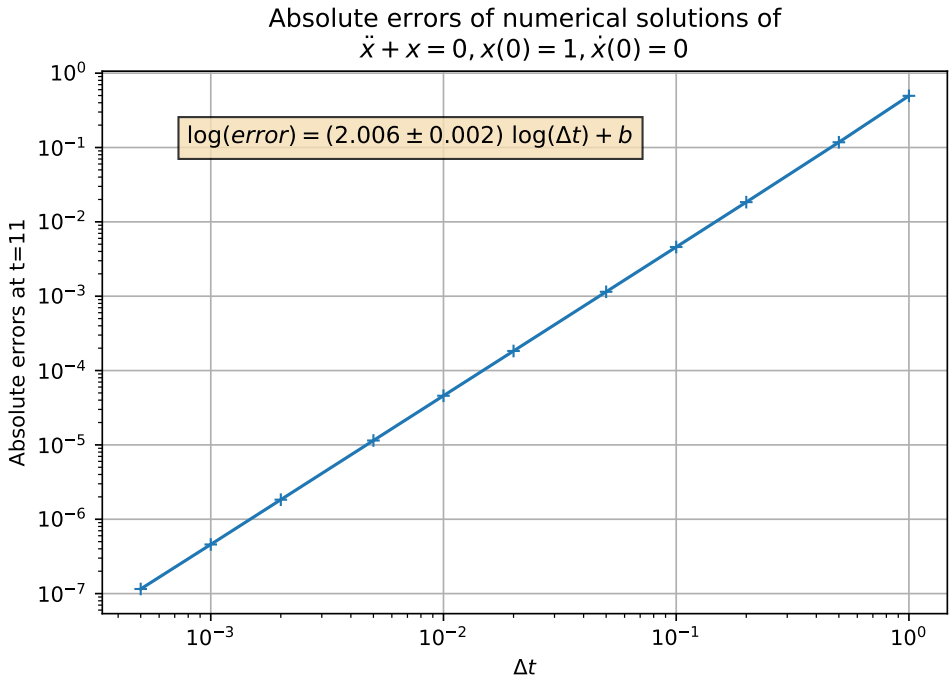

Figure 5: Dependence of absolute errors of numerical solution of the problem $\ddot{x} - x = 0, x(0) = 1, \dot{x}(0) = 0$ on time step.

We can see from Figure 5 that there is a strong increasing linear relationship between the logarithms of the error and the time step. In order to calculate its slope, we find an equation for the line of best fit using *linregress* function from the *scipy* package:

$$\log(\text{error}) = (2.006 \pm 0.002)\log(\Delta t) + b,$$

where b is some value for the y-intercept. We can see that the slope is within three standard errors from 2, and the standard error is smaller than 0.001 of the slope value. Therefore, relation between the maximum error and the time step is, approximately:

$$\text{Global error} \sim \Delta t^2.$$

Hence, we conclude that our numerical solution method is of the order $O(\Delta t^2)$.

Errors for very small Δt

We have found that errors are proportional to Δt^2. We want to see if this remains true for very small values of Δt, such as:

$$\Delta t = 0.0002, 0.0001, 0.00001, 0.000001, 0.0000001.$$

The corresponding errors are shown on Figure 6. We can see that proportionality

$$\text{Global error} \sim \Delta t^2.$$

no longer holds for $\Delta t < 0.0005$. Is we make Δt smaller than 0.0005, the errors start to increase, not decrease. The increase of errors could be caused by the rounding errors of the floating point numbers and arithmetic operations with those numbers. These errors accumulate with each time step and may become significant for small values of Δt and large number of time steps.

Our conjecture could be tested by modifying the Fortran program and using quadruple precision for real variables instead of the currently used double precision. If the errors for very small Δt are indeed caused by the limitations of the floating point number system, then we expect that quadruple precision will decrease the errors for $\Delta t = 0.0002, 0.0001, 0.00001$ and maintain the proportionality

$$\text{Global error} \sim \Delta t^2.$$

.

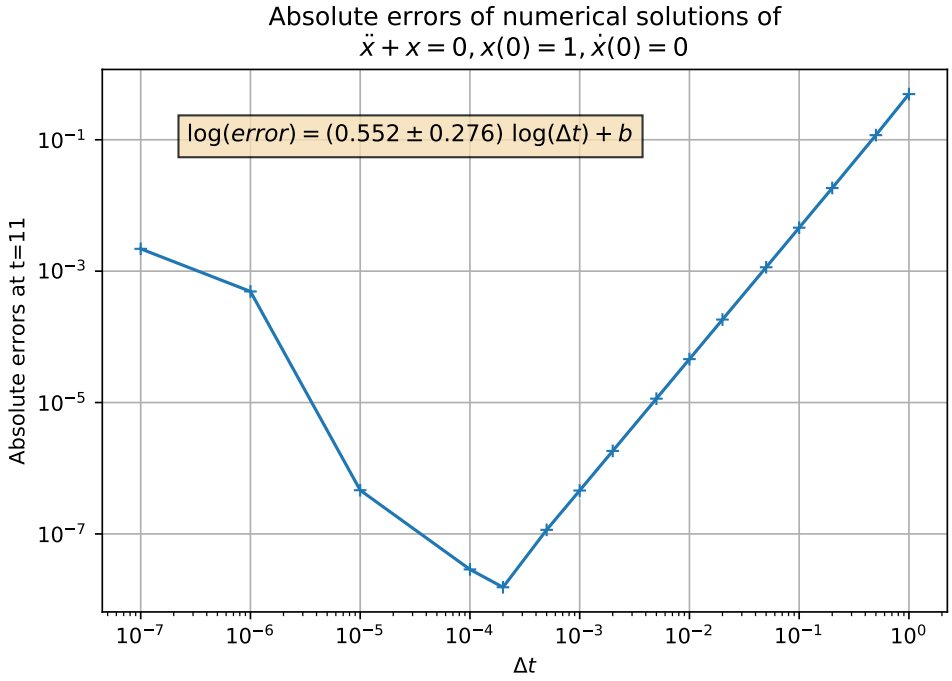

Figure 6: Dependence of absolute errors of numerical solution of the problem $\ddot{x} - x = 0, x(0) = 1, \dot{x}(0) = 0$ on time step including $\Delta t < 0.0005$.

Conclusion

We wrote a Fortran program for solving the $\ddot{x} - x = 0, x(0) = 1, \dot{x}(0) = 0$ problem numerically. We analyzed the absolute errors of our numerical approximation of the solution. We found that global errors of our numerical method were proportional to the square of the time step when using the time steps between 0.0005 and 1. Time steps smaller than 0.0005 caused the errors to increase.